VZ LINK magazine

ISSUE 45 JULY 1990

New Zealand VZ 300/200 Computer users club

FROM ME TO YOU

Well here we are again, it sure doesn't take long for the next issue to full due.

I hope you all have had time to get some programming done, now that the long cold nights are with us.

I am still short of items for use in this newsletter, if you have anything please send it in.

I thank all those that have supplied material in the past. It is hard to believe that this club is nearly 4 years old. I sure hope I can keep the club and newsletter going.

My wife now has her VZ set up in Te Kauwhata and I look forward to some more software reviews from her. Hint Hint.

THIS ISSUE

Once again we have another very fine program written by Ken Clarke. We are proud to introduce to you Donkey Kong.
Enter and play you will be glad you did.

Micro Magic this issue takes us into the use of MODE(1), It will help to explain the mysterious world of SET, RESET, and POINT. We also have a game called MICROMAZE which is part of the Micro Magic series.

We also have a item which should of appeared in the November 1989 issue. It tells you how to add three function keys to your VZ. Thanks to Ken Clarke for bringing this to my attention.

Another MailList Enhancement from Pete Patterson helps to complete this issue.

NEXT ISSUE

Well unfortunately the contents of the August issue is still a mystery, even to me. So it will be a surprise to all of us.

********** **********

SPECIAL VZ USERS MEETING

We will be holding another VZ users workshop on:-

SATURDAY 28 JULY 1990
Starting at 10-30 AM

At the RAILWAY SOCIAL CLUB ROOMS, BEACH ROAD AUCKLAND CITY.

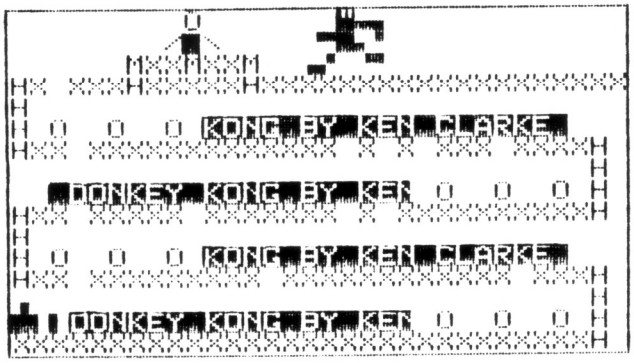
This meeting is a special one, It is your opportunity to meet and talk to BOB KITCH.

So make the effort, I am sure you will be glad you did.

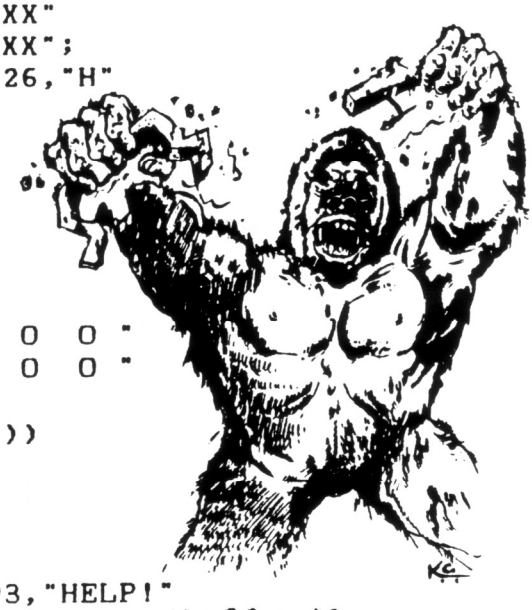
********** **********

MURPHY'S LAWS. (1)

IF ANYTHING CAN GO WRONG IT WILL.
WHEN ALL ELSE FAILS, READ THE INSTRUCTIONS.
WHEN THE PLANE YOU ARE ON IS LATE, THE PLANE YOU WANT TO TRANSFER TO IS ON TIME.




```
1 COLOR5,1:POKE30771,1:CLS
5 REM "--> DONKEY KONG <-- BY KEN CLARKE ..... MAR. 1980."
10 S$="  DONKEY KONG BY KEN CLARKE "
20 FORS=160TO448STEP96:PRINT@S,S$;:NEXTS
100 C=1:D=21:F=-1:E=1:A=14:B=2:G=0:S=0:M=0
140 PRINT@96,"XX XXXHXXXXXHXXXXXXXXXXXXXXXXXX"
150 PRINT@192,"XXX XXXXXXXXXXXXX X X XXX XXXXX"
160 PRINT@288,"XXX XXXXX XXXXXXX X XXXXXXXXXXX"
170 PRINT@384,"XXX XXXXXXXXX XXXXXXXXXXXX XXXX"
180 PRINT@480,"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
190 FORN=96TO192STEP32:PRINT@N,"H":PRINT@N+126,"H"
200 PRINT@N+192,"H":PRINT@N+318,"H";
210 NEXTN
220 PRINT@70,"MXXMXXM"
230 PRINT@9,"O          ▛▜          : gTW
240 PRINT@40,"/ \       ▗ ▜▟  gZ  : gYZSF
250 PRINT@78," ▛▗ ▟▗                : gQGsER
260 PRINT@416,"▛":PRINT@448," ▌"  : gQ : ZI
270 PRINT@160+C," O  O  O ":PRINT@256+D," O  O  O "
280 PRINT@352+C," O  O  O ":PRINT@448+D," O  O  O "
290 PRINT@A*32+B,;
300 A$=CHR$(PEEK(PEEK(30752)+256*PEEK(30753)))
310 PRINT@A*32+B," ▌"              : gI
320 IFA$="O",9000
330 C=C+E:D=D+F
340 PRINT@15,"  ▛▜                 : gTW
350 PRINT@78," ▛▗ ▟▗":PRINT@48,"▗ ▜▟":PRINT@3,"HELP!"
360 PRINT@(A+1)*32+B,;                          : sgQGsgER
370 B$=CHR$(PEEK(PEEK(30752)+256*PEEK(30753)))
380 IFASC(B$)=96,600
390 IFC=21,E=-1
400 IFC=2,E=1
410 IFD=2,F=1
420 IFD=21,F=-1
430 G=G+1:M=INT(G/60):S=G-M*60
440 IFA=1ANDB=9,9080
450 PRINT@A*32+B,A$
455 JS=(INP(42)AND31):K=(INP(37)AND31)
460 B=B-(INKEY$=","ANDB<30)+(INKEY$="M"ANDB>0)
465 IFJS=27ANDB>0,B=B-1ELSEIFJS=23ANDB<30,B=B+1
470 IFINKEY$="."ANDA$="H",GOSUB6000
475 IFJS=15,GOSUB7000ELSEIFK=15,GOSUB8000
476 IFJS=30ANDA$="H",GOSUB6000
480 IFASC(A$)=77,PRINT@A*32+B,"W":A=A-1:PRINT@A*32+B," ▌"
500 IFINKEY$="X",GOSUB7000
520 IFINKEY$="Z",GOSUB8000
530 PRINT@15,"▛▜▌  ":PRINT@78," ▟▗ ▛▗ ":PRINT@46," ▟▛ ▗ "
540 PRINT@3,"      "
550 GOTO270
600 PRINT@A*32+B," ":PRINT@(A+1)*32+B,"▌"
```



NOTE.

Graphic Characters above are shown by "g" plus the Key. Spaces are shown by "s"

Graphics below should follow the same pattern.

```
610 PRINT@(A+2)*32+B,;
620 G$=CHR$(PEEK(PEEK(30752)+256*PEEK(30753)))
630 IFG$<>" ",A=A+1
640 IFG$<>" ",9000
650 A=A+1
6000 FORN=ATOA-2STEP-1
6010 PRINT@N*32+B,"▮"
6020 FORJ=1TO3
6030 NEXTJ
6040 PRINT@N*32+B,"H"
6050 NEXTN
6060 PRINT@(N-1)*32+B,"▮":PRINT@(N-1)*32+B," "
6070 A=A-3
6080 RETURN
7000 PRINT@A*32+B," ":PRINT@(A-1)*32+B,"▖":PRINT@(A-1)*32+B," "
7010 PRINT@(A-1)*32+B,"▮":PRINT@(A-1)*32+B," "
7020 PRINT@(A-1)*32+B+1,"▮":PRINT@(A-1)*32+B+1," "
7030 PRINT@(A-1)*32+B+2,"▖":PRINT@(A-1)*32+B+2,"▮"
7040 PRINT@(A-1)*32+B+2," ":PRINT@(A-1)*32+B+2,"▖"
7050 PRINT@(A-1)*32+B+2," ":PRINT@A*32+B+2,;
7060 U$=CHR$(PEEK(PEEK(30752)+256*PEEK(30753)))
7070 IFU$="O",9000
7080 PRINT@A*32+B+2,"▮":PRINT@A*32+B+2," "
7090 B=B+2
7100 RETURN
8000 PRINT@A*32+B," ":PRINT@(A-1)*32+B,"▖":PRINT@(A-1)*32+B," "
8010 PRINT@(A-1)*32+B,"▮":PRINT@(A-1)*32+B," "
8020 PRINT@(A-1)*32+B-1,"▮":PRINT@(A-1)*32+B-1," "
8030 PRINT@(A-1)*32+B-2,"▖":PRINT@(A-1)*32+B-2," "
8040 PRINT@(A-1)*32+B-2,"▮":PRINT@(A-1)*32+B-2," "
8050 PRINT@A*32+B-2,;
8060 U$=CHR$(PEEK(PEEK(30752)+256*PEEK(30753)))
8070 IFU$="O",9000
8080 PRINT@A*32+B-2,"▮":PRINT@A*32+B-2," "
8090 B=B-2
8100 RETURN
9000 FORN=1TO25
9010 PRINT@A*32+B,"▮":PRINT@A*32+B,"▟"
9020 PRINT@A*32+B,"▮":PRINT@A*32+B,"▜"
9030 NEXTN
9040 PRINT@234,"GAME OVER"
9050 FORQ=1TO1000:NEXT
9060 CLS
9070 GOTO100
9080 PRINT@60,"▖▜▖":PRINT@93,"▟"
9090 FORN=1TO1000:NEXTN
9100 PRINT@58,"▖▜▜▖ ":PRINT@91,"▟▮   "
9110 PRINT@0,"                        ":PRINT@40,"                    "
9120 PRINT@36,"TIME:";:PRINTUSING"##:##";M,S
9130 PRINT@78,"                ":PRINT@20,"▜▜\O/":PRINT@50,"  ▖ ▟▟▟"
9140 PRINT@82,"▜▖ ▖▜▜":PRINT@25,"HELP!"
9150 FORU=1TO1500:NEXTU:CLS:GOTO100
```

Donkey Kong is a great game of rescuing a '**damsel** in distress' from the clutches of **KING KONG**. You control a little man and you have to climb up ladders and along girders to rescue the princess.

But unfortunately all sorts of things seem to get in your way, including barrels which roll along the girders. There are several holes in the girders which you must leap over otherwise you fall through with stunning effects. If you manage to reach the princess, a Love Heart appears but is cracked as Kong grabs the princess back into his clutches. Don't be scared by the beating of kong's chest, otherwise you may lose your concentration.

CONTINUED ON NEXT PAGE

Here is a tip on jumping over the barrels.
Wait till the barrel is directly in front or behind you before you attempt to leap over it. Be quick on the button as the next barrel rolls toward you.
Lastly, make sure you do not hang around near the end of the screen, otherwise you may be hit as the barrels change direction.

Controls:- 'M' - Left ', ' - Right

'Z' - Jump left 'X' - Jump right

'. ' - Climb

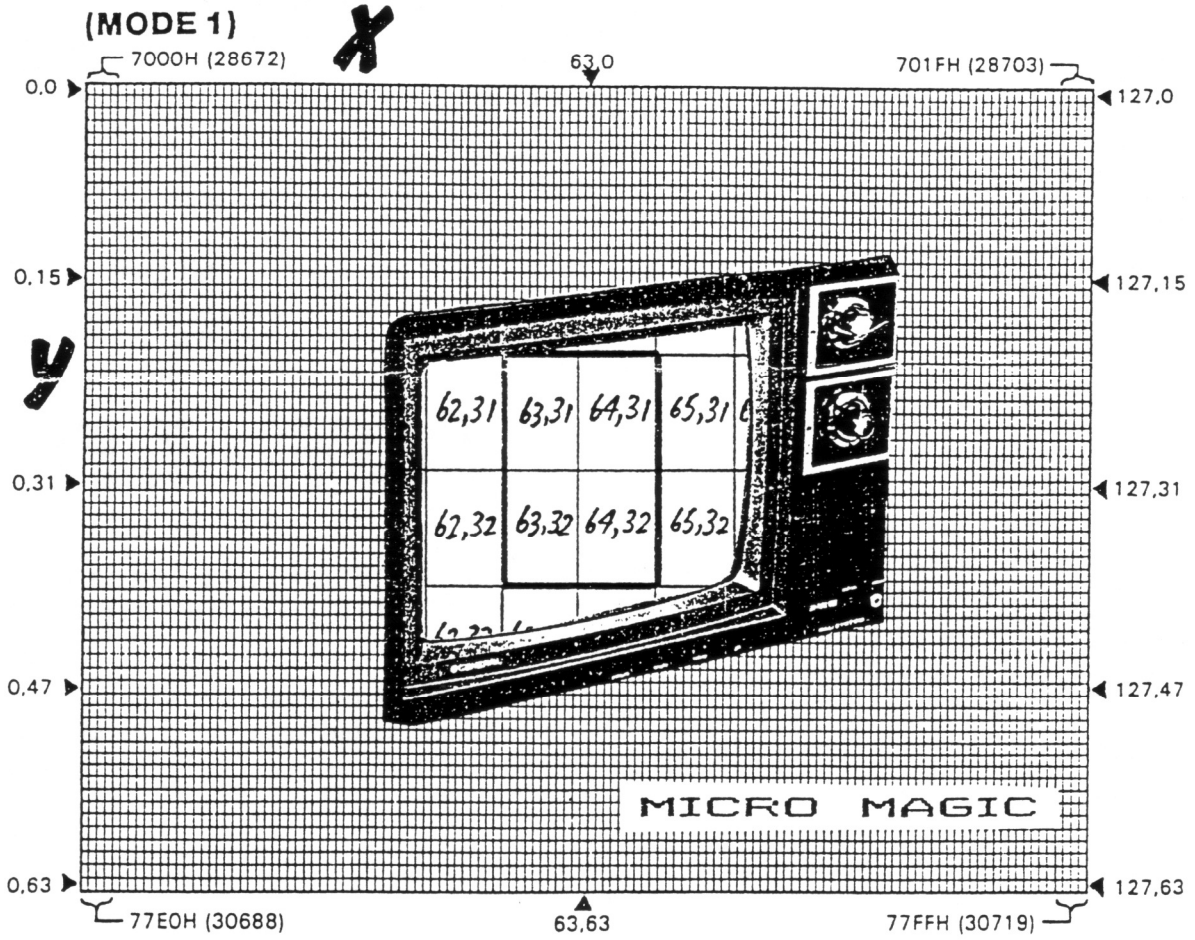
JOYSTICK CONTROL:- Either Joystick or the Keyboard may be used, and you can change from one to the other while the game is still in progress.

(REMEMBER DO NOT PLUG in joysticks while VZ is switched on)

The **'FIRE'** and **'ARM'** Buttons are used for jump right and left respectively. If other joysticks without the Arm facility are used, then you should make the changes in lines 455, 465, 475, 476.
If you experience any problems with getting this program up and running, then don't hesitate to contact me at a WORKSHOP MEETING or leave a message with peter.

MICRO MAGIC
BEGINNER'S GUIDE TO BASIC

MODE(1)



RUN and READ...

This **MICRO MAGIC** takes a different form.
VeeZed BASIC programmers must be familiar with the High Resolution mode, **MODE(1)**, and there is no substitute to *"HANDS-ON"* learning.
So, instead of my writing thousands of words and confusing you, a few *example* programs should illustrate MODE(1) much better.
VeeZedders! Start your engines!

PIXELS: PIXEL is an abbreviation for picture cell. There are 8192 of them in MODE(1). Ø to 63 (64) ACROSS and Ø to 127 (128) DOWN. See the MODE(1) Screen Grid above.

CO-ORDINATES: Like a map, PIXELS are identified by co-ordinates. In the above Screen Grid you will see an X at the top (horizontal Pixels) and a Y at the side (vertical Pixels). If X=Ø and Y=Ø they would be the co-ordinates for the top, left-hand pixel. If X=127 and Y=63 they would be the co-ordinates for the bottom, right-hand pixel. The TV acts as a magnifying-glass for the co-ordinates of the grid in the centre of the screen.

COLOR: To draw a line, a shape or a dot on the MODE(1) screen we have to change the color of the pixel(s). And this is where we go to "HANDS-ON" examples. Type in PROGRAM A.

A

SET(X,Y)

```
10 MODE(1)
20 C=1:Y=5:GOSUB100
30 C=2:Y=10:GOSUB100
40 C=3:Y=15:GOSUB100
50 C=4:Y=20:GOSUB100
60 C=5:Y=25:GOSUB100
70 C=6:Y=30:GOSUB100
80 C=7:Y=35:GOSUB100
90 C=8:Y=40:GOSUB100
92 GOTO92
100 COLORC,0:FORX=10TO50:SET(X,Y):NEXT:RETURN
```

RUN and READ...
In each of lines 20 to 90 C is the COLOR.
Y is the Vertical Co-ordinate.
In the sub-routine (line 100) the loop will draw a line across the screen. The 1st and 5th line cannot be seen because they are the same color as the backgraound.
Notice that COLORs 2, 3 and 4 are the SAME as 6, 7 and 8.
Now, change the COLOR command in line 100 to read COLORC,1.
RUN and READ...
The background has changed from GREEN to BUFF but still COLORs 2, 3 and 4 are the same as 6, 7 and 8.
Following is a Table of available COLORs:

COLOR TABLE	Pixel Color	Backgroound Color
COLORC,0	GREEN	GREEN
2,0	YELLOW	GREEN
3,0	BLUE	GREEN
4,0	RED	GREEN
1,1	BUFF	BUFF
2,1	LT. BLUE	BUFF
3,1	MAGENTA	BUFF
4,1	ORANGE	BUFF

Now BREAK and LIST100
Erase RETURN
Type the following line 110...

RESET(X,Y)

```
110 FORX=10TO40:RESET(X,Y):NEXT:RETURN
```

Notice that X=10TO40 whereas line 100 is X=10TO50.

RUN and READ
SET(X,Y) changes the COLOR of the Pixels.
RESET(X,Y) changes the COLOR of the Pixels back to the BACKGROUND COLOR.

POINT(X,Y)

POINT(X,Y) returns a number (1 to 4) indicating the color of a pixel.

B

NEW and enter PROGRAM B.

```
10 MODE(1)
20 COLOR7,1
30 SET(20,30)
40 A=POINT(20,30)
50 FORI=1TO1000:NEXT
60 PRINTA
70 GOTO70
```

RUN and READ
POINT(20,30) returns number 3 indicating the COLOR of the spot. Change line 20 to read COLOR3,1 and you will see why we do not use 5, 6, 7 and 8. They are the same as 1, 2, 3 and 4 anyway.

And now for the big one. Ha! Ha! There's nothing like jumping in the deep end. We'll see...

NEW
Type

```
10 MODE(1):COLOR3,0:GOSUB1000
20 COLOR2:X=0:Y=32:SET(X,Y)
30 A$=INKEY$:A$=""
40 A$=INKEY$:IFA$=""THEN40
50 IFZ$<>"B"THEN70ELSEX=X+1
60 SET(X,Y):RESET(X-1,Y)
70 A=POINT(X+1,Y)
80 IFA=3THENPRINT@103,"CRASH!"
90 GOTO40

1000 X=63:FORY=0TO63:SET(X,Y):NEXT:RETURN
```

RUN and READ

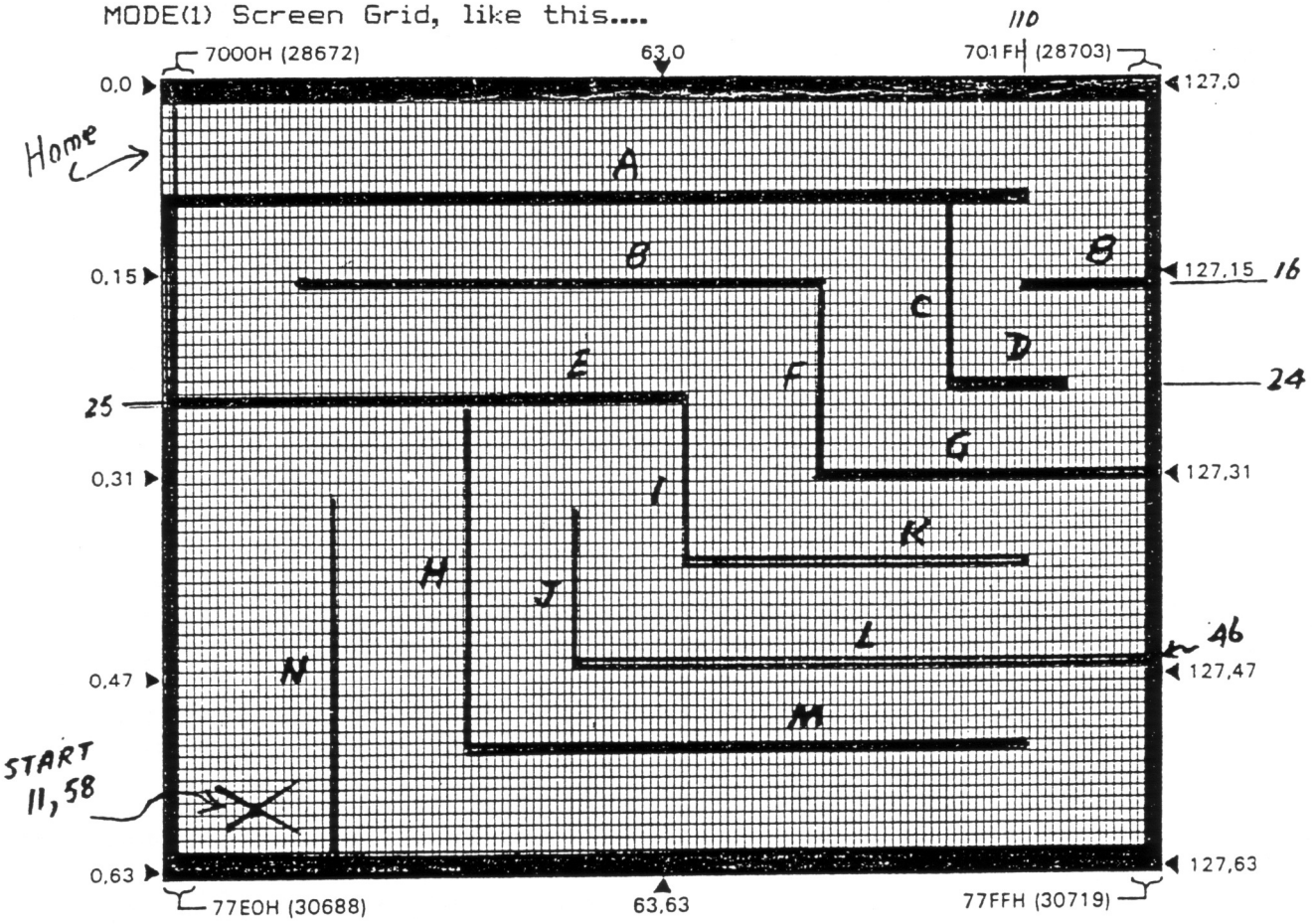
After MODE(1) and COLOR3,0 sub-routine 1000 draws a line down the centre of the screen. The COLOR of the line is 3.
RETURN brings us back to line 20 where the COLOR is changed to 2 for the rest of the program. The now familiar INKEY$ causes the computer to wait until a key is pressed. Only a B is accepted. When B is pressed X is increased by 1.
Then SET(X,Y) in line 60 will put a spot next to the first one.
RESET wipes out the first one making it appear as though the spot has moved along.
A is made equal to the COLOR just ahead of the spot by POINT(X+1,Y).
LINE 80 says, if the COLOR of the pixel just one place ahead of the spot is 3 then PRINT "CRASH".
LINE 90 will keep going back to line 40 until the "CRASH!".

This is a very simple example of MODE(1) MOVING GRAPHICS.
Without POINT the spot would pass through the line.
Try putting SOUND31,1 in line 60. You will find out how even the shortest SOUND can slow down a program like this.

MICROMAZE

Now for a MODE(1) program. We begin with a blank screen, a blank cassette and a blank expression. Where to start? Firstly we need an idea. We have in mind a MAZE in which the player must steer a path to HOME, but if he hits the wall, or his own tail, he CRASHES. Also extra hazards will appear at random and to hit one of these is SUDDEN DEATH! Points are scored as he goes with a BONUS of 1000 if he reaches HOME.

The elements of such a program are few and do not demand a FLOWCHART, so long as we keep in mind the necessity for keeping the ACTION part of the program free from anything that would slow the action down. The INSTRUCTIONS and DRAW MAZE sub-routines will need to be AFTER the game. We plan to make the drawing of the maze begin at line 1000. The INSTRUCTIONS can begin at, say, line 1200. However, we must start by drawing the maze on a MODE(1) Screen Grid, like this....



HERE WE GO THEN...

```
100 CLS:PRINT@73,"MICROMAZE":PRINT:PRINT:Z$=INKEY$:Z$=""
110 PRINT@129,"INSTRUCTIONS? (Y OR N)"
120 Z$=INKEY$:IFZ$=""THEN120
130 IFZ$<>"N"THENGOSUB1200
140 MODE(1):COLOR,1:GOSUB1000
```

Let's get the INSTRUCTIONS out of the way

```
1199 'INSTRUCTIONS
1200 CLS:PRINT@1,"MICROMAZE.  STEER YOURSELF BY"
1202 PRINT" USING THE FOLLOWING KEYS...":PRINT
1204 PRINTTAB(12)"W = UP"
1206 PRINT:PRINT"  A = LEFT                S = RIGHT"
1208 PRINTTAB(12)Z - DOWN"
1210 PRINT:PRINT" HIT THE WALL OR DRIVE INTO"
1212 PRINT" YOUR OWN TAIL AND YOU CRASH!"
1214 PRINT" HAZARDS WILL APPEAR ON THE"
1216 PRINT" TRACK AS YOU GO. IT'S SUDDEN"
```

```
1218 PRINT" DEATH TO HIT ONE. POINTS ARE"
1220 PRINT" EARNED WITH AN EXTRA BONUS"
1222 PRINT" OF 1000 IF YOU ARRIVE HOME."
1224 PRINT"        GOOD LUCK!!!"
1230 FORT=1TO6000:NEXT:SOUND31,2:RETURN
```

In line 1206 there are 2 SPACES before A and 12 SPACES before S.

Now comes the hard part. We have to enter the sub-routine for drawing the MODE(1) maze, translating from our Screen Grid. This sub-routine will need two sub-routines of its own. One to SET the HORIZONTAL (X) maze lines, and another to SET the VERTICAL (Y) maze lines. The DRAW MAZE sub-routine is to begin at line 1000, but we'll jump ahead to lines 1100 and 1200 first to enter the drawing subs.

```
1100 FORX=ATOB:SET(X,Y):NEXT:RETURN'    HORIZONTAL LINES
1110 FORY=ATOB:SET(X,Y):NEXT:RETURN'      VERTICAL LINES
```

ENTER Temporary line: 1090 GOTO1090 to allow TEST RUNS.
Lines 1100 and 1110 will draw the maze lines depending on what values we give A, B and X for the Horizontal lines, and what values we give A, B and Y for the Vertical lines.
A and B is the length of the X lines and uses FORX=ATOB, while A and B is the length of the Y lines and uses FORY=ATOB.
OK! Let's draw the maze...

```
 999 'DRAW MAZE
1000 COLOR2:A=0:B=127:Y=0:GOSUB1100:Y=Y+1:GOSUB1100'TOP BORDER
```

RUN - Check that the TOP BORDER is correct - then LIST-1090.

```
1002 Y=62:GOSUB1100:Y=Y+1:GOSUB1100'                BOTTOM BORDER
```

TEST RUN - OK? - LIST-1090.
Continue entering and testing each line in turn.
The REM letters refer to the letters on the Screen Grid.

```
1004 A=9:B=61:X=0:GOSUB1110:X=X+1:GOSUB1110'        LEFT BORDER
1006 A=2:X=126:GOSUB1110:X=X+1:GOSUB1110'          RIGHT BORDER
1008 A=2:B=110:Y=9:GOSUB1100'                                    A
1010 A=17:B=82:Y=17:GOSUB1100:A=110:B=125:Y=16:GOSUB1100'        B
1012 A=10:B=24:X=100:GOSUB1110'                                  C
1014 A=101:B=115:Y=24:GOSUB1100'                                 D
1016 A=2:B=66:Y=25:GOSUB1100'                                    E
1018 A=18:B=30:X=82:GOSUB1100'                                   F
1020 A=82:B=125:Y=31:GOSUB1100'                                  G
1022 A=26:B=53:X=39:GOSUB1110'                                   H
1024 A=26:B=37:X=66:GOSUB1110'                                   I
1026 A=34:B=45:X=53:GOSUB1110'                                   J
1028 A=66:B=110:Y=38:GOSUB1100'                                  K
1030 A=53:B=125:Y=46:GOSUB1100'                                  L
1032 A=39:B=110:Y=54:GOSUB1100'                                  M
1034 A=33:B=61:X=21:GOSUB1110'                                   N
1036 COLOR4:FORY=2TO8:SET(0,Y):NEXT'                             HOME
1038 COLOR3:SET(11,58)
1040 RETURN
```

You will have noticed that the LEFT BORDER has A=9 leaving a gap between 2 and 8. Line 1036 fills that gap in a different COLOR. This will be used to let us know if the player reaches HOME. Line 3038 marks the spot from where the player starts his journey.

COLOR 1 is the BACKGROUND.
COLOR 2 is the MAZE WALLS.
COLOR 3 is the MOVING SPOT.
COLOR 4 is the HOME.

ERASE Temporary line 1090.
LIST 140 and ADD, after GOSUB 1000 :X=11:Y=58

As mentioned earlier, there are only 4 colors. The background uses one which leaves us with only three to play with. This is limiting as we really needed another one. In the next few lines we will be responding to four events. Reaching HOME, hitting the wall, hitting one's own tail AND crashing into the random hazards. But with only three possible POINT numbers, two of the events will have to share the same response.

NOW would be a good time to **SAVE** all that hard slogging, eh??

```
200 Z$=INKEY$:Z$=""
210 Z$=INKEY$:IFZ$=""THEN210
220 IFZ$="W"THENY=Y-1:GOTO300
230 IFZ$="Z"THENY=Y+1:GOTO300
240 IFZ$="A"THENX=X-1:GOTO300
250 IFZ$="S"THENX=X+1:GOTO300
260 GOTO210
```

We now have a program that RUNs but the keys for playing don't work yet....but, the next five lines will fix that...

```
300 IFPOINT(X,Y)=2THENPRINT@96,"CRASH!  YOU'RE DEAD!":GOTO500
310 IFPOINT(X,Y)=3THENPRINT@96,"YOU HIT YOUR OWN TAIL!!!":GOTO500
320 IFPOINT(X,Y)=4THENQ=1000:PRINT@96,"BONUS 1000!!!":GOTO500
330 COLOR2:A=RND(125):B=RND(61):SET(A,B):COLOR3
340 SET(X,Y):SC=SC+1:GOTO210
```

We put the checking before SET(X,Y) so that we could check the COLOR before it was changed. To check AFTER the SET would need a rather complicated system of finding out which way the player had moved. By checking the new SET before the move it MUST be correct. Agree?

Line 330 SETs RANDOM dots in COLOR2 and then puts the COLOR back to 3. We considered putting in an extra line to prevent the RANDOM dots being SET on maze lines. Something like - IFPOINT(A,B)=2THEN330 but this would have to come before the SET(A,B).
Line 340 SETs the new position and increases SC, which is the SCORE by 1 each time a move is made.

And that only leaves us from line 500 to the end

```
500 PRINT@160,"YOUR SCORE IS "SC+Q
510 Z$=INKEY$:Z$=""
520 PRINT@224,"ANOTHER GAME? (Y OR N)"
530 Z$=INKEY$:UZ$=""THEN530
540 IFZ$="N"THENCLS:END
550 IFZ$="Y"THEN140
560 GOTO530
```

EXTRA KEYS

This issue we present a guide on how to add three function keys to the VZ200/300.

The basic VZ has three gaps in the keyboard matrix. These are unused, and not recognised by the BASIC interpreter, but the user can write interrupt driven routines which scan the relevant row addresses and carry out some special function if the key is pressed. In a previous issue we have showed how to implement a 'clear screen ' key. The possibilities are enormous - you can define a key to LIST or RUN, or carry out a disk catalogue. Function keys are built in standard on many other micros.

The function keys take the positions as shown in the table below:

ROW ADDRESS:	BIT 2	'PIN No.'
	Pin No. 14	
68FEH (26878)	F1	3
68F7H (26871)	F2	4
68EFH (26863)	F3	5

NOTE that all function keys are on column 4 of the matrix, which corresponds to bit 2.

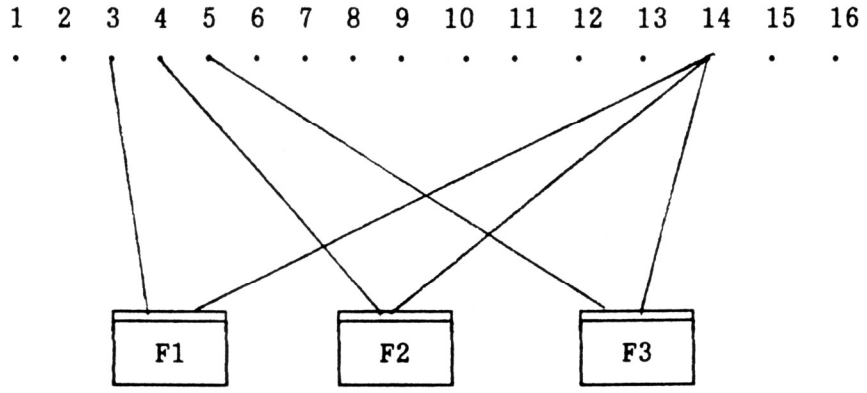
The 'Pin numbers' are needed to identify the connection points on the keyboard itself. Each key can be any simple momentary contact switch you wish. Push to make, release to break type, these are small and easy mounted.

Locating the three keys is up to you. I suggest placing them in a row down the left-hand side of the keyboard, drilling into the main case of the computer. There is just enough clearance here for the switches. You may want to mount them on the top panel of the case; just make sure there is adequate clearance when the case is closed.

We'll consider the VZ200 first. After removing the keyboard from the main assembly, carefully unscrew the miniature screws from the keyboard (don't lose any!). You should now be able to see the soldering joints where the ribbon cable meets the keyboard. Holding the board with the solder at the top and facing you, number the connections from left to right.

On the VZ300, remove the keyboard from the main case, and, noting the solder connections along the bottom right, number these from left to right.

Using which ever system you have, connect the keys as shown below:



Refer to the OCTOBER 89 issue for further information.

(Reprinted with permission of Mark Harwood from his VZ USER magazine)

MAILLIST ENHANCEMENT

A FURTHER ENHANCEMENT OF THE UPDATED MAILING LIST PROGRAM

by Pete Patterson.

At the conclusion of any disc drive operation, the program tells you to **"OPEN DRIVE DOOR"**, but there is no prompt to close the drive door.
After a couple of frustrating losses of data, I added the following prompt to the mailing list.
Line 51 is the reminder string.
Additions to line 775 prints the string with a sound change. This line is used for both loading and saving data.
Line 863 clears the screen of the CF$ at the commencement of the load file operation.
Additions to line 898 clears the screen of the CF$ at the commencement of the save file operation.
Addition to line 640 deletes CF$ if QUIT option is selected.

```
51 CF$="[illegible]"
640 PRINT@354,SP$:PRINT@386,SP$:PRINT@450,SP$:PRINT@418,SP$
755 PRINT@450,SQ$;:PRINT@420,CF$;:SOUND30,1;31,1;30,1
863 PRINT@418,SP$
898 PRINT@418,SP$;:PRINT@450,SP$;:SOUND30,1:IFUP=0THEN906
```

BUY SELL SWAP

WANTED TO BUY OR SWAP

VZ printer interface, will buy or swap for Ram pack.
Contact Peter Hill ph 761-451

FOR SALE

Seikosha GP 100A Dot Matrix printer.Tandy format.
(This is the original printer DSE recommended and sold for the VZ)

$150.00 contact Pete Patterson Phone 818-4735 after 6pm.

P.O BOX 1972 C.P.O AUCKLAND N.Z

(C) AUCKLAND VZ 300/200 COMPUTER USERS CLUB (1986)
NEW ZEALAND VZ 300/200 COMPUTER USERS CLUB (1989)

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT.
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT IT IS FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR ANY BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OR USE OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME IF YOU HAVE PROBLEMS.
PHONE 761-451 AFTER 6.00 PM.

Each article in this series is a gentle introduction to some topic in the computing field. It is written for the beginner, so may appear very simple to the rest of you. If you find it too easy, and so not worth reading, congratulations — you are a beginner no more! Each issue will deal with a different topic, of general interest. Occasionally, material may seem to repeat what has already appeared in Bits & Bytes — but remember, new readers are coming along all the time.

Sorting out data

By Gordon Findlay

What is data? How is it stored in a computer? How can a computer handle names and addresses as opposed to numbers?

Most people are fairly comfortable with the idea of a computer handling numbers. After all, that's what computers were invented for. But a lot of the time computers seem to be dealing with items of information which aren't numbers at all. I have just received a letter from the tax department's computer. Lots of numbers on it, sure — but there's my name, and my address. And there's a date — "7 Feb 1985" — not a number although it does appear to contain numbers

Any sort of information is called data. There are basically two types: numeric data, consisting of numbers such as measurements, prices, quantities and amounts; and non-numeric such as names, addresses, dates, part numbers, and so on.

Confusion can arise with some "numbers" which are classified as non-numeric.The address "12 Main St" includes a number; my computer has the product number, "3003", on it. But no good could come of adding, multiplying or subtracting those numbers. Nor does it make sense to "add" part number 12345 to part number 23456. These part numbers are just identifying strings of symbols.

Computers are able to handle numbers. But how can they handle names? Ultimately, all data is encoded numerically. The most common such code, at least in the world of micros, is called ASCII — American Standard Code for Information Interchange. In this code, A is represented as the number 65, B as 66, and so on. Every symbol, including punctuation marks, upper and lower case letters, and even sometimes graphical symbols such as the hearts, spades and so on of a card pack, has its own number. There is room for 256 different symbols to be given a number, so we won't run out.

In this system, "Gordon" is represented internally as "71 79 82 68 79 78". I'm ignoring the fact these numbers will be stored in binary — that would make the discussion too complex. Somewhere in the computer's internal instructions, there will be a mechanism for displaying characters on the screen, and this mechanism (the "character generator") knows that when a 65 is sent to it, the dots on the screen should be arranged in the shape of an A.

Even though the ASCII code is supposedly a standard, no two machines use exactly the same version of the

Every one's a bit different

code. For example, most machines use the codes from 0 to 127 only. The TRS80 uses higher codes, from 128 up, to represent some graphic displays. Under some circumstances, the Apple uses the codes from 128 to 255, in others the codes from 0 to 127. There are disagreements about the ordering within the code too — which should come first, upper or lower case letters? Machines differ.

Once the information or data is encoded, the computer is able to handle it, just as any other set of numbers. But of course, it must not be allowed to try to take the square root of my name. Programming languages help by forcing programmers to decide whether a variable being manipulated is a true number, or a string. This is done in BASIC for example by putting a dollar sign at the end of a variable name which is to be a string. In Pascal, the declaration will contain the word STRING or CHAR. Any attempt to multiply strings will then result in an error being reported. The report usually is "TYPE MISMATCH ERROR" or similar.

Strings are indicated in programs by using quote marks around them. "3003" is a string, 3003 without the quotes is a number, and these two are quite different. In machine code, where the programmer doesn't have the protection of a programming language to check on what he is doing, it is all too easy to confuse the two and get the wrong answer.

Data often has its own structure. Here are four pieces of data:
"Gordon Findlay",
"Bits and Bytes",
"P.O. Box 827",
"Christchurch."

Obviously they are not unrelated. Taken separately, we have four items of information here — four strings. Taken together, we have one piece of information, a name and address.

Records, fields and files

A group of items such as these form a record. Each record is one item. Within each record there are, in this case, four parts, called fields. Each field may be manipulated separately, or the whole record taken as one.

Records may also be grouped together, into files. As an example, let's take a telephone directory.

The directory is a file. Not only that, it is a sorted file, in that the records are in a particular order, designed to help find any individual one.

Each record relates to one person or firm. The record has four fields, the name, the address, the exchange and the number:

Bits & Bytes,	P.O. Box 827,	Chch,	66566,	
first	second	3rd	4th	fields one record.

In more complex sets of data, fields may be nested within each other. In some bridge tournaments, each game in a round will have the same hands dealt at every table. We need to be able to store the complete details of a deal:

ONE COMPLETE DEAL

East	West	North	South	
				<— a record
				<— each hand is a field.

Spades	AKJ82	
Hearts	10632	each hand has a field for each suit,
Clubs	74	which contains the details of the cards
Diamonds	J5	

Here, each record has four fields, each of which has four sub-fields, containing the actual information.

Describing such a data structure can be very easy or very difficult, depending on the programming language used. In many of the most sophisticated languages such as Pascal, C and so on, a deal can be handled and manipulated just as one item, in the sort of language used ordinarily. In BASIC and many other of the less sophisticated languages, a deal must be represented as an array (or list) of numbers, which increases the difficulty of writing, understanding, and debugging the program immensely.